A Bayesian Network Model for Interesting Itemsets

A Bayesian Network Model for Interesting Itemsets

Jaroslav Fowkes (✉) and Charles Sutton

School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK
{jfowkes,csutton}@inf.ed.ac.uk

Abstract. Mining itemsets that are the most interesting under a statistical model of the underlying data is a commonly used and well-studied technique for exploratory data analysis, with the most recent interestingness models exhibiting state of the art performance. Continuing this highly promising line of work, we propose the first, to the best of our knowledge, generative model over itemsets, in the form of a Bayesian network, and an associated novel measure of interestingness. Our model is able to efficiently infer interesting itemsets directly from the transaction database using structural EM, in which the E-step employs the greedy approximation to weighted set cover. Our approach is theoretically simple, straightforward to implement, trivially parallelizable and retrieves itemsets whose quality is comparable to, if not better than, existing state of the art algorithms as we demonstrate on several real-world datasets.

1 Introduction

Itemset mining is one of the most important problems in data mining, with applications including market basket analysis, mining data streams and mining bugs in source code [1]. Early work on itemset mining focused on algorithms that identify all itemsets which meet a given criterion for pattern quality, such as all *frequent itemsets* whose support is above a user-specified threshold. Although appealing algorithmically, the list of frequent itemsets suffers from *pattern explosion*, i.e., is typically long, highly redundant and difficult to understand [1]. In an attempt to address this problem, more recent work focuses on mining *interesting itemsets*, smaller sets of high-quality, non-redundant itemsets that can be examined by a data analyst to get an overview of the data. Several different approaches have been proposed for this problem. Some of the most successful recent approaches, such as MTV [19], KRIMP [28] and SLIM [26] are based on the *minimum description length* (MDL) principle, meaning that they define an encoding scheme for compressing the database based on a set of itemsets, and search for the itemsets that best compress the data. These methods have been shown to lead to much less redundant pattern sets than frequent itemset mining.

In this paper, we introduce an alternative, but closely related, viewpoint on interesting itemset mining methods, by starting with a probabilistic model of the data rather than a compression scheme. We define a *generative model* of the data, that is, a probability distribution over the database, in the form

of a Bayesian network model, based on the interesting itemsets. To infer the interesting items, we use a probabilistic learning approach that directly infers the itemsets that best explain the underlying data. Our method, which we call the *Interesting Itemset Miner* (IIM)[1], is to the best of our knowledge, the first generative model for interesting itemset mining.

Interestingly, our viewpoint has a close connection to MDL-based approaches for mining itemsets that best compress the data (Section 3.9). Every probability distribution implicitly defines an optimal compression algorithm, and conversely every compression scheme implicitly corresponds to a probabilistic model. Explicitly taking the probabilistic modelling perspective rather than an MDL perspective has two advantages. First, focusing on the probability distribution relieves us from specifying the many book-keeping details required by a lossless code. Second, the probabilistic modelling perspective allows us to exploit powerful methods for probabilistic inference, learning, and optimization, such as submodular optimization and structural expectation maximization (EM).

The collection of interesting itemsets under IIM can be inferred efficiently using a structural EM framework [9]. One can think of our model as a probabilistic relative of some of the early work on itemset mining that formulates the task of finding interesting patterns as a covering problem [11,28], except that in our work, the set cover problem is used to identify itemsets that cover a transaction *with maximum probability*. The set cover problem arises naturally within the E step of the EM algorithm. On real-world datasets we find that the interesting itemsets seem to capture meaningful domain structure, e.g. representing phrases such as *anomaly detection* in a corpus of research papers, or regions such as *western US states* in geographical data. Notably, we find that IIM returns a much more diverse list of itemsets than current state of the art algorithms (Table 2), which seem to be of similar quality. Overall, our results suggest that the interesting itemsets found by IIM are suitable for manual examination during exploratory data analysis.

2 Related Work

Itemset mining was first introduced by Agrawal and Srikant [2], along with the Apriori algorithm, in the context of market basket analysis which led to a number of other algorithms for frequent itemset mining including Eclat and FPGrowth. Frequent itemset mining suffers from *pattern explosion*: a huge number of highly redundant frequent itemsets are retrieved if the given minimum support threshold is too low. One way to address this is to mine *compact representations* of frequent itemsets such as maximal frequent, closed frequent and non-derivable itemsets with efficient algorithms such as CHARM [31]. However, even mining such compact representations does not fully resolve the problem of pattern explosion (see Chapter 2 of [1] for a survey of frequent itemset mining algorithms).

An orthogonal research direction has been to mine *tiles* instead of itemsets, i.e., subsets of rows *and columns* of the database viewed as binary transaction

[1] https://github.com/mast-group/itemset-mining

by item matrices. The analogous approach is then to mine *large tiles*, i.e., sub-matrices with only 1s whose area is greater than a given minimum area threshold. The Tiling algorithm [11] is an example of an efficient implementation that uses the greedy algorithm for set cover. Note that there is a correspondence between tiles and itemsets: every large tile is a closed frequent itemset and thus algorithms for large tile mining also suffer from pattern explosion to some extent.

In an attempt to tackle this problem, modern approaches to itemset mining have used the *minimum description length* (MDL) principle to find the set of itemsets that best summarize the database. MTV [20] uses MDL coupled with a *maximum entropy* (MaxEnt) model to mine the most informative itemsets. MTV mines the set of top itemsets with the highest likelihood under the model via an efficient convex bound that allows many candidate itemsets to be pruned and employs a method for more efficiently inferring the model itself. Due to the partitioning constraints necessary to keep computation feasible, MTV typically only finds in the order of tens of itemsets, whereas IIM has no such restriction.

KRIMP [28] employs MDL to find the subset of frequent itemsets that yields the best lossless compression of the database. While in principle this could be formulated as a set cover problem, the authors employ a fast heuristic that does not allow the itemsets to overlap (unlike IIM) even though one might expect that doing so could lead to better compression. In contrast, IIM employs a set cover framework to identify a set of itemsets that cover a transaction with highest probability. The main drawback of KRIMP is the need to mine a set of frequent itemsets in the first instance, which is addressed by the SLIM algorithm [26], an extension of KRIMP that mines itemsets directly from the database, iteratively joining co-occurring itemsets such that compression is maximised.

The MaxEnt model can also be extended to tiles, here known as the *Rasch* model, and, unlike in the itemset case, inference takes polynomial time. Kontonasios and De Bie [16] use the Rasch model to find the most surprising set of *noisy tiles* (i.e., sub-matrices with predominantly 1s but some 0s) by computing the likelihood of tile entries covered by the set. The inference problem then takes the form of weighted budgeted maximum set cover, which can again be efficiently solved using the greedy algorithm. The problem of Boolean matrix factorization can be viewed as finding a set of frequent noisy tiles which form a low-rank approximation to the data [22].

The MINI algorithm [10] finds the itemsets with the highest surprisal under statistical independence models of items and transactions from a precomputed set of closed frequent itemsets. OPUS Miner [29] is a branch and bound algorithm for mining the top *self-sufficient* itemsets, i.e., those whose frequency cannot be explained solely by the frequency of either their subsets or of their supersets.

In contrast to previous work, IIM maintains a generative model, in the form of a Bayesian network, *directly* over itemsets as opposed to indirectly over items. Existing Bayesian network models for itemset mining [14,15] have had limited success as modelling dependencies between the items makes inference for larger datasets prohibitive. In IIM inference takes the form of a weighted set cover problem, which can be solved efficiently using the greedy algorithm (Section 3.3).

The structure of IIM's statistical model is similar to existing models in the literature such as Rephil ([24], §26.5.4) for topic modelling and QMR-DT [25] for medical diagnosis. Rephil is a multi-level graphical model used in Google's AdSense system. QMR-DT is a bi-partite graphical model used for inferring significant diseases based on medical findings. However, the main contribution of our paper is to show that a binary latent variable model can be useful for selecting itemsets for exploratory data analysis.

3 Interesting Itemset Mining

In this section we will formulate the problem of identifying a set of interesting itemsets that are useful for explaining a database of transactions. First we will define some preliminary concepts and notation. An *item* i is an element of the universe $U = \{1, 2, \ldots, n\}$ that indexes database attributes. A *transaction* X is a subset of the universe U and an *itemset* S is simply a set of items i. The set of interesting itemsets $\mathcal{I}$ we wish to determine is therefore a subset of the power set (set of all possible subsets) of the universe. Further, we say that an itemset S is *supported* by a transaction X if $S \subseteq X$.

3.1 Problem Formulation

Our aim in this work is to infer a set of interesting itemsets $\mathcal{I}$ from a database of transactions. By *interesting*, we mean a set of itemsets that will best help a human analyst to understand the important properties of the database, that is, interesting itemsets should reflect the important probabilistic dependencies among items, while being sufficiently concise and non-redundant that they can be examined manually. These criteria are inherently qualitative, reflecting the fact that the goal of data mining is to build human insight and understanding. In this work, we formalize interestingness as those itemsets that best explain the transaction database under a *statistical model* of itemsets. Specifically we will use a *generative* model, i.e., a model that starts with a set of interesting itemsets $\mathcal{I}$ and from this set generates the transaction database. Our goal is then to infer the most likely generating set $\mathcal{I}$ under our chosen generative model. We want the model to be as simple as possible yet powerful enough to capture correlations between transaction items. A simple such model is to iteratively sample itemsets S from $\mathcal{I}$ and let their union form a transaction X. Sampling S from $\mathcal{I}$ uniformly would be uninformative, but if we associate each interesting itemset $S \in \mathcal{I}$ with a probability π_S, we can sample the indicator variable $z_S \sim \text{Bernoulli}(\pi_S)$ and include S in X if $z_S = 1$. We formally define this generative model next.

3.2 Bayesian Network Model

We propose a simple directed graphical model for generating a database of transactions $X^{(1)}, \ldots, X^{(m)}$ from a set $\mathcal{I}$ of interesting itemsets. The parameters of our model are Bernoulli probabilities π_S for each interesting itemset $S \in \mathcal{I}$. The generative story for our model is, independently for each transaction X:

1. For each itemset $S \in \mathcal{I}$, decide independently whether to include S in the transaction, i.e., sample

$$z_S \sim \text{ Bernoulli}(\pi_S).$$

2. Set the transaction to be the set of items in all the itemsets selected above:

$$X = \bigcup_{S|z_S=1} S.$$

Note that the model allows individual items to be generated multiple times from different itemsets, e.g. *eggs* could be generated both as part of a breakfast itemset {*bacon, eggs*} and as as part of a cake itemset {*flour, sugar, eggs*}.

Now given a set of itemsets $\mathcal{I}$, let $\mathbf{z}, \boldsymbol{\pi}$ denote the vectors of z_S, π_S for all $S \in \mathcal{I}$. Assuming $\mathbf{z}, \boldsymbol{\pi}$ are fully determined, it is evident from the generative model that the probability of generating a transaction X is

$$p(X, \mathbf{z}|\boldsymbol{\pi}) = \begin{cases} \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1 - z_S} & \text{if } X = \bigcup_{z_S=1} S, \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

3.3 Inference

Assuming the parameters $\boldsymbol{\pi}$ in the model are known, we can infer $\mathbf{z}$ for a specific transaction X by maximizing the posterior distribution $p(\mathbf{z}|X, \boldsymbol{\pi})$ over $\mathbf{z}$:

$$\max_{\mathbf{z}} \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1 - z_S} \qquad \text{s.t. } X = \bigcup_{S|z_S=1} S. \tag{2}$$

Taking logs and rewriting (2) in a more standard form we obtain

$$\begin{aligned} &\min_{\mathbf{z}} \sum_{S \in \mathcal{I}} z_S \left(-\ln(\pi_S)\right) + (1 - z_S) \left(-\ln(1 - \pi_S)\right) \\ &\text{s.t.} \sum_{S|i \in S} z_S \geq 1 \quad \forall\, i \in X, \qquad z_S \in \{0, 1\} \quad \forall\, S \in \mathcal{I} \end{aligned} \tag{3}$$

which is (up to a penalty term) the weighted set-cover problem (see e.g. [17], §16.1) with weights $w_S \in \mathbb{R}^+$ given by $w_S := -\ln(\pi_S)$. This is an NP-hard problem in general and so impractical to solve directly in practice. It is important to note that the weighted set cover problem is a special case of minimizing a linear function subject to a submodular constraint,[2] which we formulate as follows (cf. [30]). Given the set of interesting itemsets $\mathcal{T} := \{S \in \mathcal{I} \,|\, S \subseteq X\}$ that support the transaction, a real-valued weight w_S for each itemset $S \in \mathcal{T}$ and a non-decreasing submodular function $f : 2^{\mathcal{T}} \to \mathbb{R}$, the aim is to find a covering $C \subset \mathcal{T}$ of minimum total weight, i.e., such that $f(\mathcal{C}) = f(\mathcal{T})$ and $\sum_{S \in \mathcal{C}} w_S$ is minimized.

[2] Note that the posterior $p(z|X)$ would not be submodular if we were to use a noisy-OR model for the conditional probabilities.

Algorithm 1 HARD-EM

Input: Set of itemsets $\mathcal{I}$ and initial probability estimates $\boldsymbol{\pi}^{(0)}$
$k \leftarrow 0$
do
$k \leftarrow k + 1$
E-STEP: $\forall X^{(j)}$ solve (3) to get $z_S^{(j)}$ $\forall S \in \mathcal{T}_j$
M-STEP: $\pi_S^{(k)} \leftarrow \frac{1}{m} \sum_{j=1}^{m} z_S^{(j)}$ $\forall S \in \mathcal{I}$
while $\|\boldsymbol{\pi}^{(k-1)} - \boldsymbol{\pi}^{(k)}\| > \varepsilon$
Remove from $\mathcal{I}$ itemsets S with $\pi_S = 0$
return $\mathcal{I}, \boldsymbol{\pi}^{(k)}$

For weighted set cover we simply define $f(\mathcal{C})$ to be the number of items in $\mathcal{C}$, i.e., $f(\mathcal{C}) := |\cup_{S \in \mathcal{C}} S|$. Note that $f(\mathcal{T}) = |X|$ by construction.

We can then approximately solve the weighted set cover problem (3) using the greedy approximation algorithm for submodular functions. The greedy algorithm builds a covering $\mathcal{C}$ by repeatedly choosing an itemset S that minimizes the weight w_S divided by the number of items in S not yet covered by the covering. In order to minimize CPU time spent solving the weighted set cover problem, we cache the itemsets and coverings for each transaction as needed.

It has been shown [4] that the greedy algorithm achieves a $\ln|X| + 1$ approximation ratio to the weighted set cover problem and moreover the following inapproximability theorem shows that this ratio is essentially the best possible.

Theorem 1 (Feige [7]). *There is no* $(1 - o(1)) \ln|X|$*-approximation algorithm to the weighted set cover problem unless* $\mathsf{NP} \subseteq \mathsf{DTIME}(|X|^{O(\log \log |X|)})$*, i.e., unless* NP *has slightly superpolynomial time algorithms.*

The runtime complexity of the greedy algorithm is $O(|X||\mathcal{T}|)$, however by maintaining a priority queue this can be improved to $O(|X| \log|\mathcal{T}|)$ (see e.g. [5]). Note that there is also an $O(|X||\mathcal{T}|)$-runtime primal-dual approximation algorithm [3], however this has an approximation order of $f = \max_i |\{S \mid i \in S\}|$, i.e., the frequency of the most frequent element, which would be worse in our case.

3.4 Learning

Given a set of itemsets $\mathcal{I}$, consider now the case where both variables $\mathbf{z}, \boldsymbol{\pi}$ in the model are unknown. In this case we can use the hard EM algorithm [6] for parameter estimation with latent variables. The hard EM algorithm in our case is merely a simple layer on top of the inference algorithm (3). Suppose there are m transactions $X^{(1)}, \ldots, X^{(m)}$ with supporting sets of itemsets $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(m)}$, then the hard EM algorithm is given in Algorithm 1. To initialize $\boldsymbol{\pi}$, a natural choice is simply the support (i.e., relative frequency) of each itemset in $\mathcal{I}$.

3.5 Inferring new itemsets

We infer new itemsets using structural EM [9], i.e., we add a candidate itemset S' to $\mathcal{I}$ if doing so improves the optimal value $\overline{p}$ of the problem (3) averaged across

Algorithm 2 STRUCTURAL-EM (one iteration)

Input: Itemsets $\mathcal{I}$, probabilities $\boldsymbol{\pi}$, optima $p^{(j)}$ of (3) $\forall\, X^{(j)}$
  Set profit $\overline{p} \leftarrow \frac{1}{m}\sum_{j=1}^{m} p^{(j)}$
  do
    Generate candidate S' using CANDIDATE-GEN
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}$, $\pi_{S'} \leftarrow 1$
    E-STEP: $\forall\, X^{(j)}$ solve (3) to get $z_S^{(j)}$ $\forall\, S \in \mathcal{T}_j$
    M-STEP: $\pi'_S \leftarrow \frac{1}{m}\sum_{j=1}^{m} z_S^{(j)}$ $\forall\, S \in \mathcal{I}$
    $\forall\, X^{(j)}$, solve (3) using $\pi'_S, z_S^{(j)}$ $\forall\, S \in \mathcal{T}_j$ to get the optimum $p^{(j)}$
    Set new profit $\overline{p}' \leftarrow \frac{1}{m}\sum_{j=1}^{m} p^{(j)}$
    $\mathcal{I} \leftarrow \mathcal{I} \setminus \{S'\}$
  while $\overline{p}' \leq \overline{p}$ {until one good candidate found}
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}$
  return $\mathcal{I}, \boldsymbol{\pi}'$

transactions. Interestingly, there is an implicit regularization effect here. Observe from (3) that when a new candidate S' is added to the model, a corresponding term $\ln(1-\pi_{S'})$ is added to the log-likelihood of all transactions that S' does not support. For large databases, this amounts to a significant penalty on candidates.

To get an estimate of maximum benefit to including candidate S', we must carefully choose an initial value of $\pi_{S'}$ that is not too low, to avoid getting stuck in a local optimum. To infer a good $\pi_{S'}$, we force the candidate S' to explain all transactions it supports by initializing $\pi_{S'} = 1$ and update $\pi_{S'}$ with the probability corresponding to its actual usage once we have inferred all the coverings. Given a set of itemsets $\mathcal{I}$ and corresponding probabilities $\boldsymbol{\pi}$ along with transactions $X^{(1)}, \ldots, X^{(m)}$, each iteration of the structural EM algorithm is is given in Algorithm 2 above.

In practice, we cache the set of candidates that have been rejected by the STRUCTURAL-EM function to avoid reconsidering them.

3.6 Candidate generation

The STRUCTURAL-EM algorithm (Algorithm 2) requires a method to generate new candidate itemsets S' that are to be considered for inclusion in the set of interesting itemsets $\mathcal{I}$. One possibility would be to use the Apriori algorithm to recursively suggest larger itemsets starting from singletons, however preliminary experiments found this was not the most efficient method. For this reason we take a slightly different approach and recursively combine the interesting itemsets in $\mathcal{I}$ with the *highest support first* (Algorithm 3). In this way our candidate generation algorithm is more likely to propose viable candidate itemsets earlier and in practice we find that this heuristic works well. We did try pruning potential itemset pairs to join using a χ^2-test, however this substantially slowed down the algorithm and barely improved the model likelihood.

In order to determine the supports of the itemsets to be combined, we store the transaction database in a Memory-Efficient Itemset Tree (MEI-TREE) [8]

Algorithm 3 CANDIDATE-GEN

Input: Itemsets $\mathcal{I}$, cached supports $\boldsymbol{\sigma}$, queue length q
  if $\nexists$ priority queue $\mathcal{Q}$ for $\mathcal{I}$ **then**
    Initialize $\boldsymbol{\sigma}$-ordered priority queue $\mathcal{Q}$
    Sort $\mathcal{I}$ by decreasing itemset support using $\boldsymbol{\sigma}$
    for all distinct pairs $S_1, S_2 \in \mathcal{I}$, highest ranked first **do**
      Generate candidate $S' = S_1 \cup S_2$
      Cache support of S' in $\boldsymbol{\sigma}$ and add S' to $\mathcal{Q}$
      if $|\mathcal{Q}| = q$ **break**
    end for
  end if
  Pull highest-ranked candidate S' from $\mathcal{Q}$
  return S'

Algorithm 4 IIM (Interesting Itemset Miner)

Input: Database of transactions $X^{(1)}, \ldots, X^{(m)}$
  Initialize $\mathcal{I}$ with singletons, $\boldsymbol{\pi}$ with their supports
  Build MEI-TREE from transaction database
  while not converged **do**
    Add itemsets to $\mathcal{I}, \boldsymbol{\pi}$ using STRUCTURAL-EM
    Optimize parameters for $\mathcal{I}, \boldsymbol{\pi}$ using HARD-EM
  end while
  return $\mathcal{I}, \boldsymbol{\pi}$

and query the tree for the support of a given itemset. A MEI-TREE stores itemsets in a tree structure according to their prefixes in a memory efficient manner. To minimize the memory usage of the MEI-TREE further, we first sort the items in order of decreasing support (as in the FPGrowth algorithm) as this often results in a sparser tree [13]. Note that a MEI-TREE is essentially an FP-tree [13] with node-compression and without node-links for nodes containing the same item. An itemset support query on the MEI-TREE efficiently searches the tree for all occurrences of the given itemset and adds up their supports (see Figure 4 in [8] for the actual algorithm). With the wide availability of 100GB+ shared memory systems, it is reasonable to expect the MEI-TREE to fit into memory for all but the largest of datasets. The queue length parameter in the CANDIDATE-GEN algorithm effectively imposes a limit on the number of iterations the algorithm can spend suggesting candidate itemsets.

3.7 Mining Interesting Itemsets

Our complete interesting itemset mining (IIM) algorithm is given in Algorithm 4. Note that the HARD-EM parameter optimization step need not be performed at every iteration, in fact it is more efficient to suggest several candidate itemsets before optimizing the parameters. As all operations on transactions in our

algorithm are trivially parallelizable, we perform the E and M-steps in both the hard and structural EM algorithms in parallel.

3.8 Interestingness Measure

Now that we have inferred the model variables $\mathbf{z}, \boldsymbol{\pi}$, we are able to use them to rank the retrieved itemsets in $\mathcal{I}$. There are two natural rankings one can employ, and both have their strengths and weaknesses. The obvious approach is to rank each itemset $S \in \mathcal{I}$ according to its probability under the model π_S, however this has the disadvantage of strongly favouring frequent itemsets over rare ones, an issue we would like to avoid. Instead, we prefer to rank the retrieved itemsets according to their *interestingness* under the model, that is the ratio of transactions they explain to transactions they support. One can think of interestingness as a measure of how necessary the itemset is to the model: the higher the interestingness, the more supported transactions the itemset explains. Thus interestingness provides a more balanced measure than probability, at the expense of missing some frequent itemsets that only explain some of the transactions they support. We define interestingness formally as follows.

Definition 1. *The* interestingness *of an itemset $S \in \mathcal{I}$ retrieved by IIM (Algorithm 4) is defined as*

$$int(S) = \frac{\sum_{j=1}^{m} z_S^{(j)}}{supp(S)}$$

and ranges from 0 *(least interesting) to* 1 *(most interesting).*

Any ties in the ranking can be broken using the itemset probability π_S.

3.9 Correspondence to existing models

There is a close connection between probabilistic models and the MDL principle [18]. Given a probabilistic model $p(X|\boldsymbol{\pi}, \mathcal{I})$ of a single transaction, by Shannon's theorem the optimal code for the model will encode X using approximately $-\log_2 p(X|\boldsymbol{\pi}, \mathcal{I})$ bits. So by finding a set of itemsets that maximizes the probability of the data, we are also finding itemsets that minimize description length. Conversely, any encoding scheme implicitly defines a probabilistic model: given an encoding scheme E that assigns each transaction X to a string of $L(X)$ bits, we can define $p(X|E) \propto 2^{-L(X)}$, and then E is an optimal code for $p(X|E)$. Interpreting previous MDL-based itemset mining methods in terms of their implicit probabilistic models provides interesting insights into these methods.

MTV uses a MaxEnt distribution over itemsets $S \in \mathcal{I}$, which for a transaction X can be written (cf. [20]):

$$p(X) = \pi_0 \prod_{S \in \mathcal{I}} \pi_S^{\mathbf{1}_X(S)}$$

where the indicator function $\mathbf{1}_X(S) = 1$ if X supports S and 0 otherwise. Thus if an itemset is present in the MaxEnt model *it must be used* to explain a supported

transaction, contrast this with IIM (1) where there is a latent variable $z_S^{(j)}$ for each transaction $X^{(j)}$ that *infers if an itemset is used* to explain the transaction.

KRIMP by contrast, uses an itemset independence model, which for an itemset $S \in \mathcal{I}$ is given by (cf. [28]):

$$p(S) = \sum_{j=1}^{m} z_S^{(j)} \Bigg/ \sum_{I \in \mathcal{I}} \sum_{k=1}^{m} z_I^{(k)}$$

where the $z_S^{(j)}$, and therefore itemset coverings for $X^{(j)}$, are determined using a *heuristic approximation*. That is, unlike IIM, the itemset coverings are not chosen to maximise the probability under the statistical model. Instead, for each transaction X, frequent itemsets $S \in \mathcal{I}$ are chosen in order of *decreasing size and support* and added to the covering if they improve the compression, until all elements of X are covered. Additionally, itemsets in the covering are not allowed to overlap, in contrast to IIM which does allow overlap if it is deemed necessary.

SLIM uses the same approach as KRIMP but iteratively finds the candidate itemsets S directly from the dataset. It employs a greedy heuristic to do this: starting with a set of singleton itemsets $\mathcal{I}$, pairwise combinations of itemsets in $\mathcal{I}$ are considered as candidate itemsets S in order of highest estimated compression gain. IIM uses a very similar heuristic that iteratively extends itemsets by the most frequent itemset in its candidate generation step (Section 3.6).

However, IIM is different from these methods in that they all contain an explicit penalty term for the description length of the itemset database, which corresponds to a prior distribution $p(\mathcal{I})$ over itemsets. We did not find in practice that an explicit prior distribution was necessary but it would be possible to trivially incorporate it. Also, if we view IIM as an MDL-type method, not only the presence of an itemset, but also its absence is explicitly encoded (in the form of $(1 - \pi_S)^{1 - z_S^{(j)}}$ in (1)). As a result, there is an implicit penalty for adding too many patterns to the model and one does not need to use a code table which would serve as an explicit penalty for greater model complexity.

One can also think of IIM as a probabilistic tiling method: each interesting itemset $S \in \mathcal{I}$ can be thought of as a binary submatrix of transactions for which $z_S = 1$ by items in S, where the choice of items and transactions in the tile are *inferred directly* from IIM's statistical model. That is, IIM formulates the inference problem (3) as a *weighted set cover* for *each transaction* where the weights correspond to *itemset probabilities*. This is in contrast to existing tiling methods: Geerts et al. [11] find k tiles covering the largest number of database entries and is thus an instance of *maximum coverage*. Kontonasios and De Bie [16] extend this to inferring a covering of noisy tiles using *budgeted maximum coverage*, that is, finding a covering that maximizes the sum of the *surprisal* of each tile, under a MaxEnt model constrained by expected row and column margins, subject to the sum of the *description lengths* of each tile being smaller than a given budget.

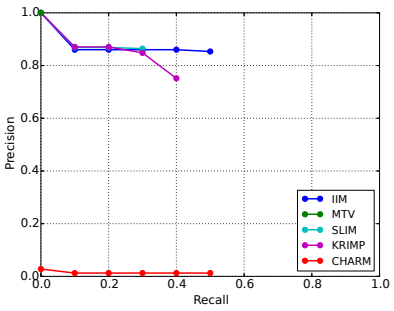

Fig. 1. Precision against recall for each algorithm on our synthetic database, using the top-k itemsets as a threshold.[3]

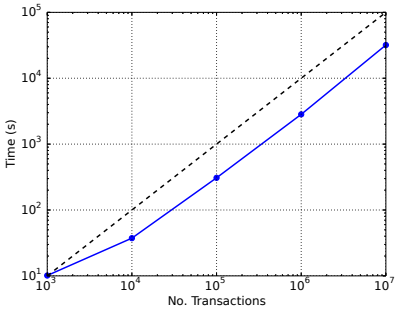

Fig. 2. IIM scaling as the number of transactions in our synthetic database increases.

4 Numerical Experiments

In this section we perform a comprehensive qualitative and quantitative evaluation of IIM. On synthetic datasets we show that IIM returns a list of itemsets that is largely non-redundant, contains few spurious correlations and scales linearly with the number of transactions. On a set of real-world datasets we show that IIM finds itemsets that are much less redundant than state of the art methods, while being of similar quality.

Datasets We use five real-world datasets in our numerical evaluation (Table 1). The plants dataset [27] is a list of plant species and the U.S. or Canadian states where they occur. The mammals dataset [23] consists of presence records of European mammals in 50×50 km geographical areas. The retail dataset consists of anonymized market basket data from a Belgian retail store [12]. The ICDM dataset [16] is a list of ICDM paper abstracts where each item is a stemmed word, excluding stop-words. The Uganda dataset consists of Facebook messages taken from a set of public Uganda-based pages with substantial topical discussion over a period of three months. Each transaction in the dataset is an English language message and each item is a stemmed English word from the message.

IIM Results We ran IIM on each dataset for $1,000$ iterations with a priority queue size of $100,000$ candidates. The runtime and number of non-singleton itemsets returned is given in Table 1 (right). We also investigated the scaling of IIM as the number of transactions in the database increases, using the model trained on the plants dataset from Section 4.1 to generate synthetic transaction databases of various sizes. We then ran IIM for 100 iterations on these databases and one can see in Figure 2 that the scaling is linear as expected. Our prototype implementation can process one million transactions in 30 seconds on 64 cores each iteration, so there is reason to hope that a more highly tuned implemen-

[3] Each curve is the 11-point interpolated precision i.e., the interpolated precision at 11 equally spaced recall points between 0 and 1 (inclusive), see [21], §8.4 for details.

tation could scale to even larger datasets. All experiments were performed on a machine with 64 AMD Opteron 6376 CPUs and 256GB of RAM.

Evaluation Criteria We will evaluate IIM along with MTV, SLIM, KRIMP and CHARM with χ^2-test ranking according to the following criteria:

1. *Spuriousness* – to assess the degree of spurious correlation in the mined set of itemsets.
2. *Redundancy* – to measure how redundant the mined set of itemsets is.
3. *Interpretability* – to informally assess how meaningful and relevant the mined itemsets actually are.

Note that we chose not to compare to the tiling methods from [11,16] as they have been shown to underperform on the ICDM dataset [20].

4.1 Itemset Spuriousness

The set-cover formulation of the IIM algorithm (3) naturally favours adding itemsets to the model whose items co-occur in the transaction database. One would therefore expect IIM to largely avoid suggesting itemsets of uncorrelated items and so generate more meaningful itemsets. To verify this is the case and validate our inference procedure, we check if IIM is able to recover the itemsets it used generate a synthetic database. To obtain a realistic synthetic database, we sampled 10,000 transactions from the IIM generative model trained on the plants dataset. We were then able to measure the precision and recall for each algorithm, i.e., the fraction of mined itemsets that are generating and the fraction of generating itemsets that are mined, respectively. We used a minimum support of 0.0575 for all algorithms (except IIM) as used in [20] for the plants dataset. Figure 1 shows the precision-recall curve for each algorithm using the top-k mined itemsets (according to each algorithm's ranking) as a threshold. One can clearly see that IIM was able to mine about 50% of the generating itemsets and almost all the itemsets mined were generating. This not only provides a good validation of IIM's inference procedure and underlying generative model but also demonstrates that IIM returns few spurious itemsets. For comparison, SLIM and KRIMP exhibited very similar behaviour to IIM whereas MTV returned a

Table 1. Summary of the real datasets used and IIM results after 1,000 iterations. † excluding singleton itemsets.

Dataset	Items	Trans.	$\lvert\mathcal{I}\rvert$†	Runtime
ICDM	4,976	859	798	163 min
Mammals	194	2,670	359	22 min
Plants	70	34,781	259	27 min
Retail	16,470	88,162	957	941 min
Uganda	33,278	124,566	928	1086 min

Table 2. IID for the top 50 non-singleton itemsets returned by the algorithms. *returned less than 50 non-singleton itemsets.

	ICDM	Mam.	Plant	Retail	Ugan.
IIM	**4.00**	**7.42**	**4.80**	**3.26**	**3.78**
MTV	3.14	*5.50	*5.00	2.52	*1.60
SLIM	2.12	*1.76	*1.77	1.44	2.08
KRIMP	2.56	1.94	1.88	1.34	2.26
CHARM	1.42	1.44	1.50	1.32	1.72

very small set of generating itemsets. The set of top itemsets mined by CHARM contained many itemsets that were not generating. It is not our intention to draw conclusions about the performance of the other algorithms as this experimental setup naturally favours IIM. Instead, we compare the itemsets from IIM with those from MTV, SLIM and KRIMP on real-world data in the next sections.

4.2 Itemset Redundancy

We now turn our attention to evaluating whether IIM returns a less redundant list of itemsets than the other algorithms on real-world datasets. A suitable measure of redundancy for a single itemset is the minimum symmetric difference between it and the other itemsets in the list. Averaging this across all itemsets in the list, we obtain the *average inter-itemset distance* (IID). We therefore ran all the algorithms on the datasets in Table 1. This enabled us to calculate, for each dataset, the IID of the top 50 non-singleton itemsets, which we report in Table 2. For CHARM, we took the top 50 non-singleton itemsets ranked according to χ^2 from the top $100,000$ frequent itemsets it returned (as the χ^2 calculation would be prohibitively slow otherwise). One can clearly see that the top IIM itemsets have a larger IID on average, and are therefore less redundant, than the KRIMP, SLIM or CHARM itemsets. The top CHARM χ^2-ranked itemsets are the most redundant as expected. On all datasets, the IIM itemsets are less redundant than those mined by the other methods, with only one exception. On the Plants dataset, MTV is slightly less redundant than IIM, but this is because MTV is unable to return 50 items on this dataset, instead returning only 21.

4.3 Itemset Interpretability

For the datasets in Table 1 we can directly interpret the mined itemsets and informally assess how meaningful and relevant they are.

ICDM Dataset We compare the top ten non-singleton itemsets mined by the algorithms in Table 3 (excluding KRIMP whose itemsets are similar for space reasons). The mined patterns are all very informative, containing technical concepts such as *support vector machine* and common phrases such as *pattern discovery*. The IIM itemsets suggest the stemmer used to process the dataset could be improved, as we retrieve {*parameter, parameters*} and {*sequenc, sequential*}.

Plants and Mammals Datasets For both datasets, all algorithms find itemsets that are spatially coherent, but as we showed in Table 2, those returned by IIM are far less redundant. Our novel interestingness measure enables IIM to rank correlated itemsets above singletons and rare itemsets above frequent ones, in contrast to the other algorithms. For example, for the plants dataset, the top itemset retrieved by IIM is {*Puerto Rico, Virgin Islands*} whereas MTV returns {*Puerto Rico*}, not associating it with the *Virgin Islands* (which are adjacent) until the 20th ranked itemset. For the mammals dataset, the top two non-singleton IIM itemsets are a group of four mammals that coexist in Scotland and Ireland and a group of ten mammals that coexist on Sweden's border with

Table 3. Top ten non-singleton ICDM itemsets as found by IIM, MTV and SLIM.

IIM	MTV	SLIM
associ rule	experiment result	inform model
local global	synthetic real	cluster algorithm
support vector machin svm	real datasets	larg effici
parameter parameters	pattern discov	perform set
anomali detect	associ rule mine	propos problem
sequenc sequential	frequent pattern mine algorithm	method set
linear discriminant analysi	train classifi	associ rule
synthetic real life	address problem	problem result
background knowledg	classifi class	approach base method
semi supervised	machin learn	base method set

Table 4. Top six non-singleton Uganda itemsets for each algorithm.

IIM	MTV	SLIM	KRIMP
soul, rest, peace	heal, jesus, amen	!, ?	whi, ?
chris, brown	god, amen	2, 4	?, !
bebe, cool	2, 4	whi, ?	2, 4
airtel, red	whi, ?	god, amen	wat, ?
everi, thing	god, heal	da, dat	time, !
time, wast	2, !	heal, jesus, amen	soul, rest, peace

Norway. By contrast, the top four SLIM and KRIMP itemsets list some of the most common mammals in Europe (see the supplementary material for details).

Uganda Dataset The top six non-singleton itemsets found by the algorithms are shown in Table 4; the IIM itemsets provide much more information about the topics of the messages than those from the other algorithms. Figure 3 (left) plots the mentions of each of the top IIM itemsets per day. As one can see, usage of the top itemsets displays temporal structure (and exhibits spikes of popularity), even though our model does not explicitly capture this. Of particular interest are the large spikes of {*soul, rest, peac*} corresponding to notable deaths: wealthy businessman James Mulwana on the 15th January, President Museveni's father on the 22nd February and six school students in a traffic accident on the 29th March. Also of interest are the 285 mentions of {*airtel, red*} on New Year's Eve corresponding to mobile provider Airtel's Red Christmas competition for 10K worth of airtime. The spike of {*bebe, cool*} on the 15th January corresponds to the Ugandan musician's wedding announcement and the spike on the 24th January of {*chris, brown*} refers to many enthusiastic mentions of the popular American singer that day. The last two itemsets capture common phrases.

In comparison, the top-six MTV itemsets are plotted in Figure 3 (right). One can see that the itemsets {*heal, jesus, amen*};{*god, amen*} and {*god, heal*} substantially overlap and are strongly correlated with each other, sharing a large spike on the 8th February and a smaller spike on the 11th March. The remaining

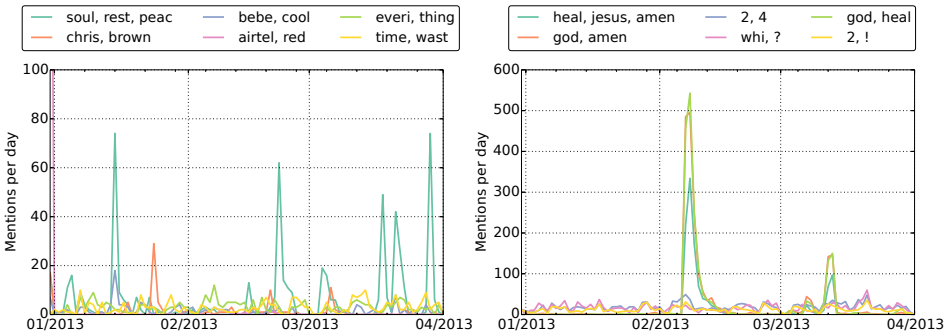

Fig. 3. Mentions per day of the top six non-singleton IIM (left) and MTV (right) itemsets from the Uganda messages dataset over three months.

itemsets exhibit no interesting spikes as one would expect. The top six SLIM and KRIMP itemsets in Table 4 all displayed random time evolution, as one would expect, except for the religious ones we have already encountered.

5 Conclusions

We presented a generative model that directly infers itemsets that best explain a transaction database along with a novel model-derived measure of interestingness and demonstrated the efficacy of our approach on both synthetic and real-world databases. In future we would like to extend our approach to directly inferring the association rules implied by the itemsets and parallelize our approach to large clusters so that we can efficiently scale to much larger databases.

Acknowledgements. This work was supported by the Engineering and Physical Sciences Research Council (grant number EP/K024043/1). We thank John Quinn for sharing the Uganda data.

References

1. Aggarwal, C., Han, J.: Frequent Pattern Mining. Springer (2014)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB. vol. 1215, pp. 487–499 (1994)
3. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. Journal of Algorithms 2(2), 198–203 (1981)
4. Chvátal, V.: A greedy heuristic for the set-covering problem. Math. O.R. 4(3), 233–235 (1979)
5. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. MIT Press (2001)
6. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B pp. 1–38 (1977)

7. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM 45(4), 634–652 (1998)
8. Fournier-Viger, P., Mwamikazi, E., Gueniche, T., Faghihi, U.: MEIT: Memory Efficient Itemset Tree for targeted association rule mining. In: Advanced Data Mining and Applications, Lecture Notes in Computer Science, vol. 8347, pp. 95–106 (2013)
9. Friedman, N.: The Bayesian structural EM algorithm. In: UAI. pp. 129–138 (1998)
10. Gallo, A., De Bie, T., Cristianini, N.: MINI: Mining informative non-redundant itemsets. In: PKDD, pp. 438–445 (2007)
11. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Discovery science. pp. 278–289 (2004)
12. Goethals, B., Zaki, M.: FIMI repository (2004), http://fimi.ua.ac.be/
13. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Record. vol. 29, pp. 1–12 (2000)
14. He, R., Shapiro, J.: Bayesian mixture models for frequent itemset discovery. arXiv preprint arXiv:1209.6001 (2012)
15. Jaroszewicz, S., Simovici, D.A.: Interestingness of frequent itemsets using Bayesian networks as background knowledge. In: SIGKDD. pp. 178–186 (2004)
16. Kontonasios, K.N., De Bie, T.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: SDM. pp. 153–164 (2010)
17. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics, Springer (2012)
18. MacKay, D.J.C.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press (2003)
19. Mampaey, M., Tatti, N., Vreeken, J.: Tell me what i need to know: succinctly summarizing data with itemsets. In: SIGKDD. pp. 573–581 (2011)
20. Mampaey, M., Vreeken, J., Tatti, N.: Summarizing data succinctly with the most informative itemsets. TKDD 6(4), 16 (2012)
21. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
22. Miettinen, P., Mielikainen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE TKDE 20(10), 1348–1362 (2008)
23. Mitchell-Jones, A., Amori, G., Bogdanowicz, W., Kryštufek, B., Reijnders, P., Spitzenberger, F., Stubbe, M., Thissen, J., Vohralík, V., Zima, J.: The Atlas of European Mammals. T & AD Poyser (1999)
24. Murphy, K.: Machine Learning: A Probabilistic Perspective. MIT Press (2012)
25. Shwe, M.A., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., Cooper, G.: Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. Methods of Information in Medicine 30(4), 241–255 (1991)
26. Smets, K., Vreeken, J.: SLIM: Directly mining descriptive patterns. In: SDM. pp. 236–247 (2012)
27. USDA: The PLANTS Database (2008), http://plants.usda.gov/
28. Vreeken, J., Van Leeuwen, M., Siebes, A.: KRIMP: mining itemsets that compress. Data Mining and Knowledge Discovery 23(1), 169–214 (2011)
29. Webb, G.I., Vreeken, J.: Efficient discovery of the most interesting associations. TKDD 8(3), 15 (2014)
30. Young, N.: Greedy set-cover algorithms (1974-1979, Chvátal, Johnson, Lovász, Stein). In: Kao, M. (ed.) Encyclopedia of Algorithms, pp. 379–381 (2008)
31. Zaki, M.J., Hsiao, C.J.: CHARM: An efficient algorithm for closed itemset mining. In: SDM. vol. 2, pp. 457–473 (2002)